

WHITEPAPER

Enterprise EKS Auto Mode Migration Roadmap

A phased plan for moving Kubernetes workloads from on-premises clusters to Amazon EKS Auto Mode, with open-source observability, security and GitOps.

PUBLISHED BY
ZSoftly Technologies Inc.

DATE
September 2026

VERSION
2.0

AUDIENCE
Engineering leaders,
platform and security
teams

CONTENTS

Executive summary	Page 2
What EKS Auto Mode changes	Page 3
Reference architecture and platform stack	Page 4
The 20-week roadmap	Page 5
Phases 1 to 7: platform build	Page 6
Phase 8: application migration	Page 7
Disaster recovery and cost	Page 8
How ZSoftly can help	Page 9
Sources	Page 10

AWS facts and prices in this paper were checked against AWS documentation on 29 September 2026. AWS changes features and prices often. Check the linked sources before you plan a budget.

EXECUTIVE SUMMARY

Let AWS run the cluster infrastructure and keep control of everything above it

Teams that run Kubernetes on their own hardware spend much of their time on work that does not ship features. They refresh servers, patch nodes, plan version upgrades and keep autoscalers, ingress controllers and storage drivers alive. Amazon EKS Auto Mode moves most of that work to AWS. This paper sets out a 20-week plan to get there without a risky cutover.

The problem

- Fixed hardware makes capacity a guess. You buy for the peak and pay for idle servers the rest of the year.
- Every Kubernetes upgrade is a project. Node patching falls behind while the team handles urgent work.
- Add-ons such as the autoscaler, load balancer controller and CSI driver each need their own upgrades and fixes.
- Visibility across workloads is limited, and commercial monitoring and security tools are priced per host.

The approach

Run each environment on an EKS Auto Mode cluster. AWS then patches and replaces the nodes, scales compute, and manages load balancing, block storage and pod networking. Put open-source tools on top: SigNoz for observability, Falco and Trivy for security, External Secrets Operator for secrets, and Argo CD for GitOps. Migrate applications one at a time behind DNS. Keep the old system running until each one proves itself.

20 weeks

Base plan for five applications, in ten phases.

One app at a time

Parallel running and a DNS rollback for every cutover.

Everything in Git

Terraform for AWS, Argo CD for the cluster.

Priced openly

A worked cost example you can check against AWS.

The outcomes to aim for

- No node operating system to patch. AWS replaces Auto Mode nodes at least every 21 days.
- Every environment rebuilt from code, so `stg` matches `prd` and recovery is a repeatable run.
- Deployments through pull requests, with Argo CD reverting drift.
- Logs, metrics, traces and runtime alerts in tools you own and control.

Who should read this

Engineering leaders should read the summary, the roadmap on page 5 and the cost model on page 8.

Platform engineers will find the architecture and phase detail on pages 3 to 6.

Application teams should read the migration method on page 7. **Security teams** should read the platform stack on page 4 and the shared responsibility table on page 3.

CHAPTER 1

What EKS Auto Mode changes

Standard EKS gives you a managed control plane and leaves the nodes to you. EKS Auto Mode extends AWS management to the nodes and to the components that make a cluster usable. AWS lists compute autoscaling, pod and service networking, application load balancing, cluster DNS, block storage and GPU support as core components rather than add-ons [1].

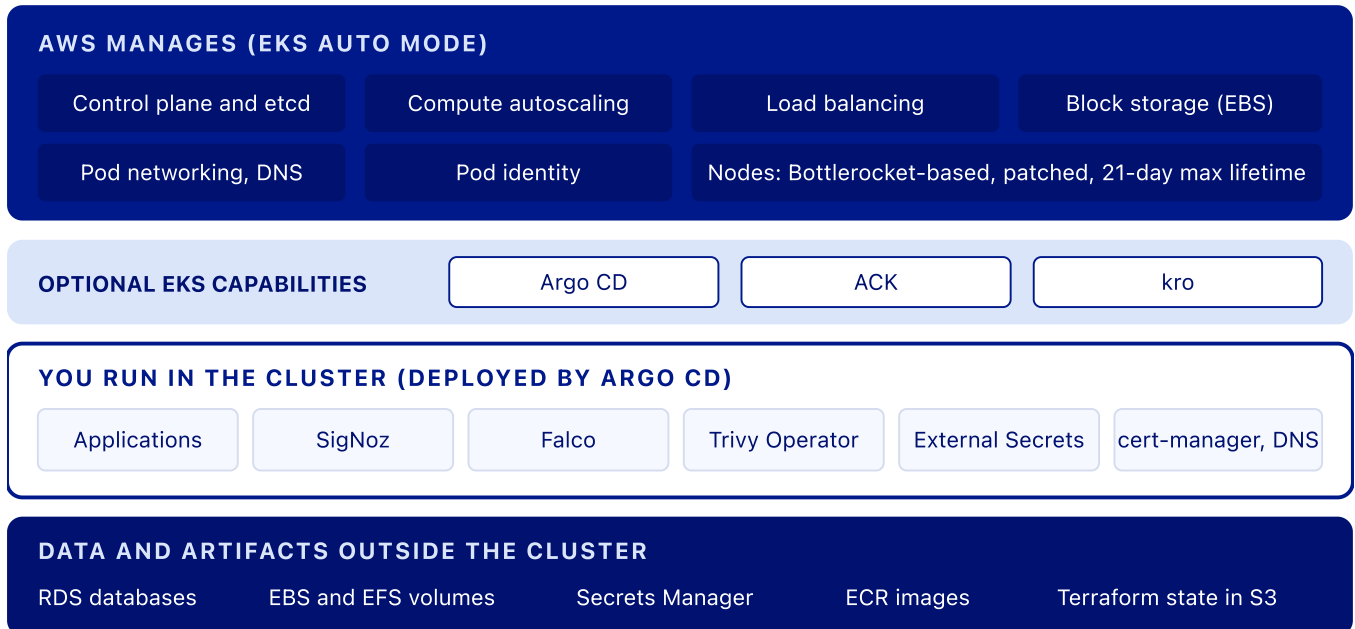


Figure 1. Who runs what. AWS operates the top band. Your team owns the cluster contents and the data layer, and keeps all of it in Git.

Area	AWS responsibility in Auto Mode	Your responsibility
Nodes	Chooses and patches the AMI, runs SELinux in enforcing mode with a read-only root file system, blocks SSH and SSM access, replaces nodes at least every 21 days [1]	Pod disruption budgets that let nodes roll, and NodePools that limit instance types
Scaling	Karpenter-based node provisioning and consolidation [1]	Pod autoscaling (HPA), resource requests, limits
Networking	Pod IP management, network policy enforcement, load balancer provisioning [1]	VPC design, security groups, the network policies themselves
Workloads	None	Containers, availability, security and monitoring of your applications [1]

Table 1. Shared responsibility under EKS Auto Mode, summarised from the AWS user guide.

Auto Mode ships two built-in NodePools. `system` carries a `CriticalAddonsOnly` taint for cluster-critical add-ons. `general-purpose` takes ordinary workloads. Both use On-Demand capacity from the C, M and R families [2]. Add your own NodePools for Spot, Graviton or GPU work. Do not edit the built-in ones.

CHAPTER 2

Reference architecture and platform stack

Accounts and regions

Use two AWS accounts under AWS Organizations. A non-production account holds `dev` and `qa`. A production account holds `stg` and `prd`, plus a disaster recovery footprint in a second region. Each environment gets its own VPC and its own cluster. A mistake in `dev` cannot reach `prd` data.

Network and cluster access

- Private subnets for nodes, public subnets only for load balancers and NAT.
- VPC endpoints for ECR, S3, STS and Secrets Manager, so image pulls and API calls stay on the AWS network.
- A private Kubernetes API endpoint. Engineers and CI runners reach it over a VPN or zero-trust access tool such as WireGuard, Tailscale or AWS Client VPN.
- EKS access entries map IAM roles to cluster permissions. Nobody shares a kubeconfig.
- Secrets encrypted with a KMS key. Control plane API, audit and authenticator logs sent to CloudWatch.
- Pick a Kubernetes version in standard support. EKS gives each version 14 months of standard support, then 12 months of extended support at a higher price [3].

GitOps: managed or self-hosted Argo CD

AWS now offers Argo CD as an EKS Capability that runs in EKS, not on your nodes [4]. It uses the same Application and ApplicationSet resources as upstream Argo CD. Check these points before you choose.

Managed capability suits you if

- You already use AWS IAM Identity Center. It is the only supported sign-in.
- You deploy only to EKS clusters. Targets are EKS cluster ARNs.
- You want AWS to handle access to private clusters.

Self-host with Helm if

- You need another SSO provider, the notifications controller or config management plugins. The capability does not support them [4].
- You deploy to clusters outside EKS.
- You need to change settings the capability locks, such as the 120-second sync timeout.

Either way, use an app-of-apps layout. One root application points at a platform folder (observability, security, secrets) and a workloads folder (your applications), with one values file per environment.

Open-source platform services

- **SigNoz** for metrics, logs and traces on OpenTelemetry, installed with its Helm chart [5]. Store its data on gp3 volumes and size them for your retention period.
- **Falco**, a CNCF graduated project, for runtime threat detection using eBPF [6]. Route alerts through Falcosekick to SigNoz and your chat tool.
- **Trivy** in the pipeline and as the Trivy Operator in the cluster, for image vulnerability and configuration reports [7].
- **External Secrets Operator** to sync values from AWS Secrets Manager into Kubernetes Secrets, so Git holds only references [8].

CHAPTER 3

The 20-week roadmap

The plan has ten phases. Phases overlap by a week where the work allows it. The base plan covers five applications. Plan roughly one to two extra weeks for each additional application, or add a second migration team for large portfolios. These are ZSoftly planning estimates. Your discovery work sets the real dates.

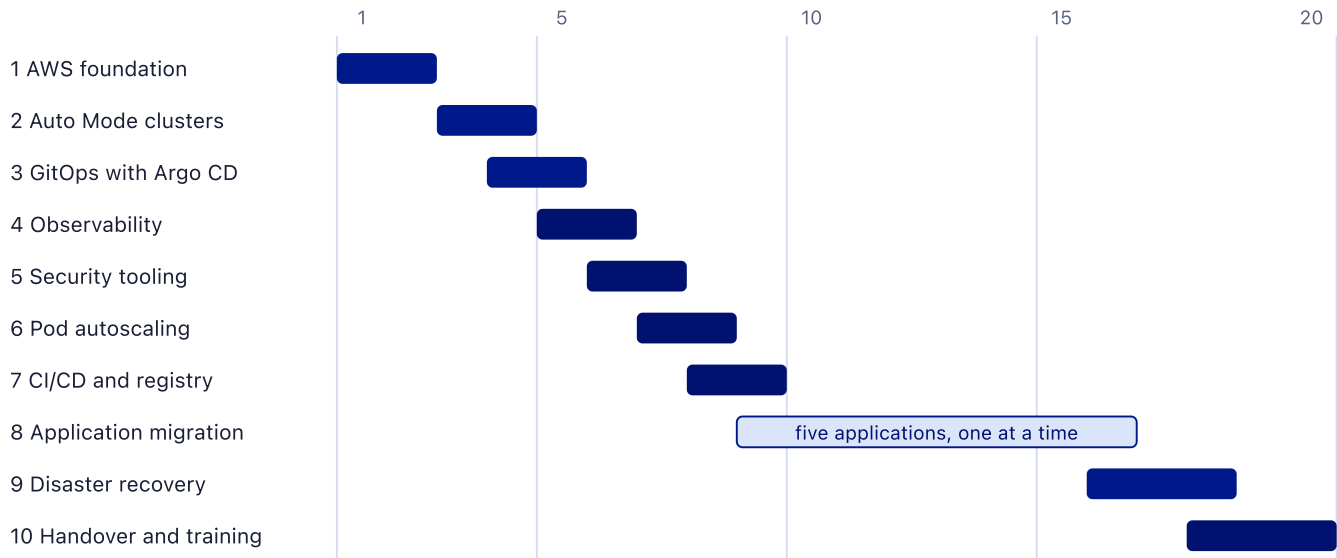


Figure 2. Phase timeline in weeks. Platform phases (1 to 7) finish before the first `prd` cutover.

#	Phase	Weeks	Exit criteria
1	AWS foundation	1 to 2	Accounts, SCPs, VPCs, endpoints, KMS keys and deploy roles created by Terraform
2	Auto Mode clusters	3 to 4	One cluster per environment, private API, access entries, logging on, custom NodePools where needed
3	GitOps with Argo CD	4 to 5	App-of-apps syncing from Git, SSO groups mapped to roles
4	Observability	5 to 6	SigNoz collecting logs, metrics and traces, first dashboards and alerts live
5	Security tooling	6 to 7	Falco alerts routed, Trivy reports in cluster, admission policy agreed
6	Pod autoscaling	7 to 8	Metrics Server running, HPA template in the standard chart
7	CI/CD and registry	8 to 9	Build, scan and push to ECR, Argo CD picks up new tags
8	Application migration	9 to 16	Five applications live on EKS, legacy instances retired
9	Disaster recovery	16 to 18	DR region defined in code and a timed failover drill passed
10	Handover and training	18 to 20	Runbooks accepted, your team runs an upgrade and a drill unaided

Table 2. Phases, durations and the evidence that closes each phase.

CHAPTER 4

Phases 1 to 7: build the platform before moving anything

Phase 1. AWS foundation

Create the account structure, SCPs and one VPC per environment with non-overlapping CIDR ranges. Leave room for pod IPs, because the VPC CNI gives each pod a VPC address. Add VPC endpoints, KMS keys and cross-account deploy roles. Write all of it in Terraform from the first day. Resources created by hand now become drift later.

Phase 2. EKS Auto Mode clusters

Create one Auto Mode cluster per environment on a Kubernetes version in standard support. Turn on API, audit and authenticator logs. Grant access through EKS access entries. Add custom NodePools only when you need them, such as Spot capacity in `dev` or GPU instances for inference work. Auto Mode only launches supported instance types larger than `small`, and it enforces IMDSv2 with a hop limit of 1 [10]. Nodes cannot be reached by SSH or SSM, so plan troubleshooting around logs, metrics and `kubectl debug`.

Phase 3. GitOps with Argo CD

Install Argo CD, managed or self-hosted, and bootstrap the app-of-apps. Map identity groups to roles. Platform admins get admin, developers get edit rights in `dev` and `qa` and read access elsewhere, and auditors get read-only. From this point, changes to the clusters go through a pull request.

Phases 4 and 5. Observability and security

Deploy SigNoz and its Kubernetes infrastructure collectors through Argo CD. Build dashboards for pod CPU and memory, restarts, request latency (p90 and p99), throughput and error rate. Then deploy Falco with the eBPF driver, starting from the default rules plus a rule for interactive shells in containers. Deploy the Trivy Operator to report critical and high findings. Run the admission check in audit mode first, and switch to blocking once teams have cleared the backlog.

Phase 6. Pod autoscaling

Auto Mode adds nodes when pods cannot be scheduled. It does not decide how many pods you need. Deploy Metrics Server and give every application chart a HorizontalPodAutoscaler with minimum and maximum replicas and a CPU target, for example 70 percent. Tune the targets in `qa` under load, not in `prd`.

Phase 7. CI/CD and container registry

Create one private ECR repository per service in Terraform, with scan on push and lifecycle rules that keep recent tagged images and expire untagged ones. For `prd`, replicate images to the DR region so a failover does not depend on the primary region. The pipeline has four steps.

- 1 Validate. Run `terraform fmt` and `validate`, lint the Helm charts and run a Trivy configuration scan.
- 2 Build and scan. Build the image and fail the job on critical findings.
- 3 Push. Tag with a semantic version and push to ECR. Never deploy `latest`.
- 4 Deploy. Update the tag in the environment's values file and let Argo CD sync it.

CI runners authenticate to AWS with OIDC and short-lived roles, never with stored access keys. Our companion paper, *CI/CD Pipelines for Infrastructure as Code*, covers that pattern in detail.

CHAPTER 5

Phase 8: migrate one application at a time

Use the strangler fig pattern. Stand each application up on EKS next to the running system, prove it, then move traffic with DNS. The old instance stays available until the new one has run in `prd` for an agreed period. Nothing moves in a single weekend.

Per-application steps

- 1 Containerise the application and write a Helm chart with one values file per environment.
- 2 Deploy to `dev` and check behaviour in SigNoz.
- 3 Promote to `qa`. Load test and tune HPA targets.
- 4 Promote to `stg` for a production-like check.
- 5 Promote to `prd` and move DNS.
- 6 Retire the on-premises instance after the validation window.

Every chart includes

- Resource requests and limits
- Liveness, readiness and startup probes
- HPA with minimum and maximum replicas
- ExternalSecret references, no secret values
- OpenTelemetry instrumentation for SigNoz
- A default-deny network policy with named exceptions
- `runAsNonRoot` and the restricted Pod Security Standard

Common pitfalls

Pitfall	What happens	Prevention
Moving the database first	Long downtime and a hard rollback	Keep the database where it is until the application is proven on EKS, then move data with replication
Ignoring connection limits	Pod scale-out exhausts database connections	Put RDS Proxy or PgBouncer in front and cap HPA maximums
High DNS TTL at cutover	Traffic keeps reaching the old system for hours	Lower the TTL to 60 seconds well before the cutover date
Stateful sets across zones	A pod cannot start because its EBS volume is in another Availability Zone	Plan volume placement and test pod rescheduling before cutover
Copying on-premises sizing	Paying for idle capacity	Profile in <code>dev</code> and <code>qa</code> , then set requests from real data
Blocking pod disruption budgets	Nodes cannot roll, so patching stalls	Allow at least one pod to be disrupted and run two or more replicas

Table 3. Migration pitfalls seen in Kubernetes moves and how to avoid them.

Rollback plan for every cutover

- 1 Before cutover, run both systems in parallel.
- 2 At cutover, point DNS at EKS and keep the old system warm.
- 3 Watch error rate and latency in `prd` for an agreed window, typically 24 to 48 hours.
- 4 If a problem appears, revert the DNS record. With a 60-second TTL, most clients move back within minutes.
- 5 Retire the old system only after the window closes cleanly.

CHAPTER 6

Phase 9: disaster recovery, and what it all costs

Cold standby in a second region

The DR region runs no workload compute day to day. Terraform modules and an Argo CD app-of-apps for the DR region sit in Git, ready to apply. ECR replicates images continuously. RDS snapshots and AWS Backup copies of EFS go to the DR region on a schedule. Keeping an empty EKS control plane there costs \$0.10 an hour and saves a step during failover.

- 1 Route 53 health checks detect the primary region failure.
- 2 Terraform creates the DR networking and cluster resources.
- 3 Argo CD syncs the DR app-of-apps from Git.
- 4 RDS restores from the latest copied snapshot.
- 5 Route 53 moves traffic to the DR endpoints after health checks pass.

Your recovery point is only as recent as the last snapshot copy. Daily copies mean up to 24 hours of data loss. If that is too much, use cross-region read replicas or continuous backups. Restore time grows with database size, so set the recovery time objective from a timed drill, not from a diagram.

Worked cost example

The table uses AWS's own EKS Auto Mode pricing example: four instances in US West (Oregon) at On-Demand rates, for 730 hours [9]. We add one cluster control plane. The Auto Mode fee is charged on top of the EC2 price. For these four instance types it equals 12 percent of the On-Demand rate.

Line item	Basis	Hourly	Monthly
EKS control plane, standard support	1 cluster × \$0.10	\$0.100	\$73.00
EC2 On-Demand	c6a.2xlarge \$0.306, c6a.4xlarge \$0.612, m5a.2xlarge \$0.344, m5a.xlarge \$0.172	\$1.434	\$1,046.82
EKS Auto Mode fee	\$0.03672 + \$0.07344 + \$0.04128 + \$0.02064	\$0.172	\$125.62
Total per month, one cluster			\$1,245.44

Table 4. Illustrative, at AWS list prices in USD, checked 29 September 2026. Check current pricing for your region before you budget.

How to read the example

- If the cluster falls into extended support, the control plane rises to \$0.60 an hour, or \$438.00 a month [9]. Upgrading on time is a cost control.
- AWS states that Auto Mode charges are independent of the EC2 purchase option [9]. Savings Plans and Spot reduce the EC2 line. Do not expect them to reduce the fee.
- The example leaves out EBS volumes, load balancers, NAT gateways, VPC endpoints, data transfer and CloudWatch. Price those from your own design.
- Weigh the fee against the engineering time you spend today on node images, patching, Karpenter and add-on upgrades.
- SigNoz, Falco and Trivy carry no licence fee. You still pay for the compute and storage they use and the time to run them.

WORKING WITH US

How ZSoftly can help

ZSoftly Technologies Inc. is a cloud engineering company. We design, build and run cloud platforms for clients in Canada, the United States and the United Kingdom. The engineers who design your platform are the ones who build it and hand it over.

Assess

Review your clusters, applications and costs. Rank applications by migration risk.

Design

Agree the account layout, network, platform stack and a dated plan in writing.

Build

Deliver the phases in this paper as code, with a demonstration at the end of each one.

Operate

Run the platform with you through the migration and the first upgrades.

Hand over

Runbooks, access and code transfer to your team, which runs a drill unaided.

Where we usually start

- 1 **Discovery call.** We review your current setup and goals. Your first hour with our engineers is free.
- 2 **Assessment.** You receive a written readiness report with an application inventory, a risk ranking and a cost estimate built from your own numbers.
- 3 **Proof of concept.** We stand up a `dev` Auto Mode cluster with SigNoz, Falco and one of your applications, so your team can judge the platform before committing.
- 4 **Delivery.** We agree the scope, price and acceptance tests for the remaining phases before work starts.

What you keep

- All Terraform, Helm charts and Argo CD configuration, in your own repositories.
- Runbooks for upgrades, incidents, scaling and disaster recovery.
- AWS accounts, keys and access that belong to you from the first day.

Check your readiness before phase 1

- Do you have an inventory of applications, their owners and their dependencies?
- Is each application already in a container, or is that part of the work?
- Where do databases live today, and how much downtime can each one take?
- Which identity provider will engineers use to reach AWS and the clusters?
- What recovery point and recovery time does the business need for each application?
- Who on your team will own the platform after handover?

If you cannot answer three or more of these, start with the assessment. It answers them in writing.

Talk to an engineer. Book a discovery call at zcp.zsoftly.ca/contact or call +1 (647) 771-1065.

REFERENCES

Sources

All sources were checked on 29 September 2026.

1. AWS. Automate cluster infrastructure with EKS Auto Mode.
<https://docs.aws.amazon.com/eks/latest/userguide/automode.html>
2. AWS. Enable or disable built-in NodePools.
<https://docs.aws.amazon.com/eks/latest/userguide/set-builtin-node-pools.html>
3. AWS. Understand the Kubernetes version lifecycle on EKS.
<https://docs.aws.amazon.com/eks/latest/userguide/kubernetes-versions.html>
4. AWS. Comparing EKS Capability for Argo CD to self-managed Argo CD.
<https://docs.aws.amazon.com/eks/latest/userguide/argocd-comparison.html>
5. SigNoz. Install SigNoz on Kubernetes.
<https://signoz.io/docs/install/kubernetes/>
6. CNCF. Cloud Native Computing Foundation announces Falco graduation, 29 February 2024.
<https://www.cncf.io/announcements/2024/02/29/cloud-native-computing-foundation-announces-falco-graduation/>
7. Aqua Security. Trivy Operator documentation.
<https://aquasecurity.github.io/trivy-operator/>
8. External Secrets Operator. AWS Secrets Manager provider.
<https://external-secrets.io/latest/provider/aws-secrets-manager/>
9. AWS. Amazon EKS pricing, including the EKS Auto Mode example.
<https://aws.amazon.com/eks/pricing/>
10. AWS. Learn about Amazon EKS Auto Mode managed instances.
<https://docs.aws.amazon.com/eks/latest/userguide/automode-learn-instances.html>

Notices

This paper is for general information. It does not create a warranty or a commitment from ZSoftly Technologies Inc. Timelines are planning estimates. Prices are AWS list prices on the date checked and will change. Amazon Web Services, AWS and Amazon EKS are trademarks of Amazon.com, Inc. or its affiliates. Kubernetes is a registered trademark of The Linux Foundation. SigNoz, Falco, Trivy, Argo CD and External Secrets Operator are open-source projects under their own licences.

© 2026 ZSoftly Technologies Inc. Version 2.0, September 2026.